智能推荐系统期末项目：
基于预训练大模型的推荐算法调研

张宏伟 10205102417

谈笑枫 10205102413

2023 年 6 月 19 日

摘要（张宏伟）

近年来，基于预训练的大型语言模型在推荐系统领域引起了广泛的关注和应用。这些模型，如 GPT、BERT 和 XLNet 等，通过在大规模文本数据上进行预训练，可以学习到丰富的语义表示和上下文理解能力。这使得它们可以在推荐系统中用于各种任务，如用户兴趣建模、商品推荐和搜索排序等。

本文的目标是对基于预训练大模型的推荐方法研究领域取得的部分成果进行总结。首先，介绍了预训练大模型的相关概念；随后，介绍了几种常见的预训练的大模型，然后对调研中学习的基于预训练大模型推荐的方法进行总结，分别是基于用户特征进行推荐、基于顺序/会话的推荐、基于预训练大语言模型进行推荐、基于双塔模型进行推荐；再选择论文对应的模型和代码进行理解、运行。

目录

1 预训练大模型介绍（张宏伟）

1.1 概念介绍

在推荐系统中，预训练大模型是指使用大规模数据集进行事先训练的深度学习模型。这些模型通常使用无监督学习的方法，例如自编码器或语言模型，对大量的文本、图像或其他类型的数据进行训练。

预训练大模型的目标是通过学习数据中的统计规律和模式来捕捉数据的表示。通过这种方式，模型可以学会从原始数据中提取有用的特征，这些特征可以用于解决各种推荐任务，如物品推荐、广告推荐、搜索排序等。

一旦预训练完成，这些模型可以通过微调或迁移学习的方式应用于具体的推荐任务。微调是指在特定任务的数据集上进一步训练模型，以适应任务的特定要求。迁移学习则是将预训练模型的知识和表示迁移到新的任务上，通常是通过调整模型的某些层或添加额外的层来完成。

1.2 基本过程

预训练大模型是一个两阶段的过程，包括预训练阶段和微调阶段。下面是预训练大模型的基本过程：

1. **数据准备：**

- 收集大规模的无监督数据集，如互联网文本、图像、音频等数据。这些数据集应该具有广泛的覆盖领域和丰富的语义信息。
- 对数据进行预处理，如分词、标注、去除噪音等，以准备输入模型的格式。

2. **构建模型：**

- 选择适当的模型架构，如 Transformer，BERT 等。这些模型通常具有多层的自注意力机制和编码器-解码器结构，用于处理输入序列的长距离依赖关系和捕捉上下文信息。
- 初始化模型的权重参数，可以使用随机初始化或者使用预训练好的权重作为初始参数。

3. **预训练阶段：**

- 使用无监督学习方法在大规模数据上进行预训练。常见的预训练任务包括：
  - 掩盖语言模型（Masked Language Modeling，MLM）：随机掩盖输入序列中的一些词汇，并要求模型预测这些被掩盖的词汇。
  - 下一句预测（Next Sentence Prediction，NSP）：给定两个连续的句子，模型需要判断它们是否是按照原文顺序相邻的。
- 使用自监督学习任务引导模型学习语言模型和上下文理解能力。通过大规模数据的无监督训练，模型可以学习到丰富的语义表示。

4. **微调阶段：**

- 准备特定任务的数据集，如推荐系统中的用户-物品交互数据、用户特征、物品特征等。
- 使用预训练的大模型作为初始参数，在特定任务的数据上进行微调。通常采用有监督学习方法，根据具体任务选择合适的损失函数和优化算法。
- 进行模型的训练和优化，更新模型的参数以最小化损失函数。微调过程中可以根据需求进行模型结构的调整和超参数的调优。

5. **应用到推荐系统：**

- 针对具体的推荐任务，可以根据需要设计模型的输出层和损失函数。

总的来说，预训练大模型的过程包括数据准备、模型构建、预训练阶段、微调阶段和应用到推荐系统等步骤。通过在大规模数据上进行预训练，模型能够学习到更丰富的语义表示，为推荐系统提供更准确和个性化的推荐结果。

1.3 优势

基于预训练的大模型在推荐系统中具有多个优势，包括：

1. 学习丰富的特征表示：预训练的大模型通过在大规模无监督数据上进行预训练，能够学习到更丰富、更有表征能力的特征表示。

2. 解决数据稀疏性问题：预训练的大模型能够通过学习大量数据的统计特征，提取出隐含的语义信息，从而缓解数据稀疏性问题，提高推荐的准确性和个性化程度。

3. 可迁移学习能力：预训练的大模型可以在不同的推荐任务和领域中进行迁移学习。降低了对大量标注数据的依赖，同时节省了训练时间和计算资源。

4. 融合多源信息：预训练的大模型能够将多源信息进行融合，包括用户行为数据、文本数据、图像数据等。可以综合考虑不同信息源的特征，并提供更全面和准确的推荐结果。
5. 可解释性和透明性：预训练的大模型通常具有较好的可解释性，可以通过注意力机制等方式，明确表示模型在推荐过程中关注的信息和权重。这有助于提高推荐系统的透明性，使用户更容易理解推荐结果的依据。

综上所述，基于预训练的大模型在推荐系统中能够提供更丰富的特征表示、解决数据稀疏性问题、具备迁移学习能力，同时融合多源信息、提供可解释性和透明性等优势，从而提升推荐系统的性能和用户体验。

2.常见的预训练大模型（张宏伟）

下面介绍在推荐系统中常见的预训练的大模型：

2.1 BERT 模型

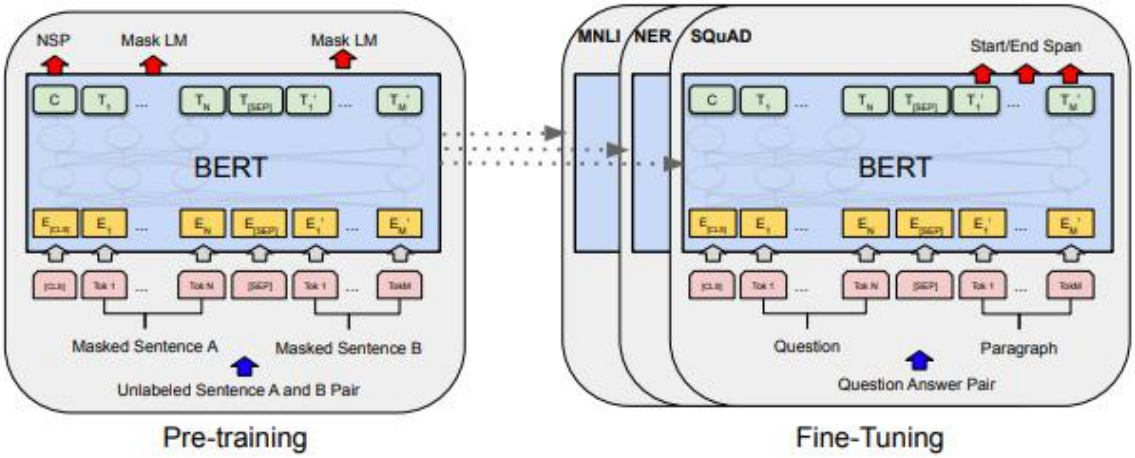
参考 BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

2.1.1 BERT 模型简介

BERT 是 2018 年 10 月由 Google AI 研究院提出的一种预训练模型。BERT 的全称是 Bidirectional Encoder Representation from Transformers。BERT 是一种基于 Transformer 结构的预训练模型，通过在大规模未标记数据上进行语言建模和遮蔽语言模型任务的预训练。BERT 在推荐系统中可以用于文本理解、相似度计算和推荐候选生成等任务。

2.1.2 BERT 的架构

BERT 整体框架包含 pre-train 和 fine-tune 两个阶段。pre-train 阶段模型是在无标注的标签数据上进行训练，fine-tune 阶段，BERT 模型首先是被 pre-train 模型参数初始化，然后所有的参数会用下游的有标注的数据进行训练。



BERT 的核心思想是利用 Transformer 的自注意力机制，实现双向的语言建模。相较于传统的单向语言模型，BERT 能够同时利用上下文的信息，更好地理解文本中的语义关系。

2.1.3 BERT 的预训练过程

BERT 的预训练过程包含两个主要任务：

语言建模 (Language Modeling)：BERT 在预训练阶段通过双向语言建模任务，学习预测输入序列中任意位置的词汇。这个任务使得 BERT 能够捕捉到上下文之间的依赖关系，从而学习到更具有语义理解能力的表示。

遮蔽语言模型（Masked Language Modeling）：BERT 在输入序列中随机遮蔽一些词或部分词，然后通过上下文中其他词的信息，预测被遮蔽的词。这个任务鼓励 BERT 学习到上下文感知的表示，能够进行词汇推理和补全。

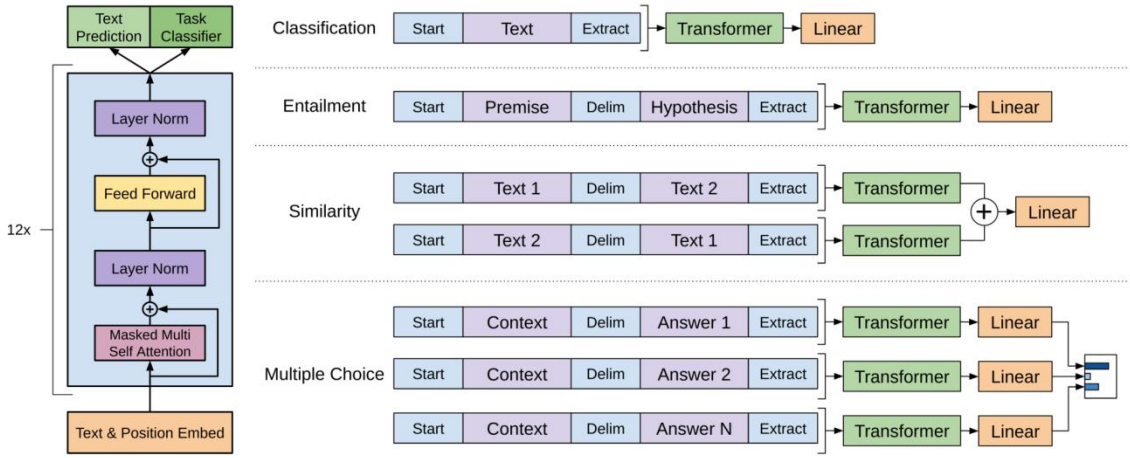
预训练完成后，BERT 的模型参数包含了大量的语义信息，可以用于下游任务的微调。在微调阶段，将特定任务的标注数据与预训练的 BERT 模型结合，通过有监督学习进行模型参数的微调，以适应特定任务的需求。

2.2 GPT 模型

参考： Improving Language Understanding by Generative Pre-training

GPT（Generative Pre-trained Transformer）是基于 Transformer 架构的预训练语言模型。它采用了无监督学习的方式，在大规模未标记的文本数据上进行预训练，学习语言的统计特征和语义表示。

GPT 模型的核心组件是 Transformer，它由多个堆叠的 Encoder Blocks 构成。每个 Encoder Block 由多个 Self-Attention 和 Feed-Forward 层组成。在 GPT 模型中，Transformer 的 Encoder 部分被称为 Decoder，因为它是一个单向语言模型。



GPT 的训练的两阶段过程：

1.无监督的预训练语言模型¶

给定句子 U = [u1, u2, ..., un]，GPT 训练语言模型时的目标是最大化下面的似然函数：

$$L_1(U) = \sum_i \log P(u_i|u_{i-k}, \ldots, u_{i-1}; \Theta)$$

有上述公式可知，GPT 是一个单向语言模型，假设输入张量用 h0 表示，则计算公式如下：

$$h_0 = UW_e + W_p$$

其中 Wp 是单词的位置编码, We 是单词本身的 word embedding. Wp 的形状[max_seq_len, embedding_dim], We 的形状是[vocab_size, embedding_dim].得到输入张量 h0 后，要将 h0 传入 GPT 的 Decoder Block 中，依次得到 ht:

$$h_t = \text{transformer\_block}(h_{l-1}), \quad l \in [1, t]$$

最后通过得到的 ht 来预测下一个单词:

$$P(u) = \text{softmax}(h_t W_e^T)$$

2.有监督的下游任务 fine-tunning

GPT 经过预训练后，会针对具体的下游任务对模型进行微调. 微调采用的是有监督学习，训练样本包括单词序列[x1, x2, ..., xn]和 label y. GPT 微调的目标任务是根据单词序列[x1, x2, ..., xn]预测标签 y.

$$(y|x_1, \ldots, x_m) = \text{softmax}(h_l^m W_y)$$

其中 Wy 表示预测输出的矩阵参数，微调任务的目标是最大化下面的函数:

$$L_2 = \sum_{(x,y)} \log P(y|x_1, \ldots, x_m)$$

综合两个阶段的目标任务函数，可知 GPT 的最终优化函数为:

$$L_3 = L_2 + \lambda L_1$$

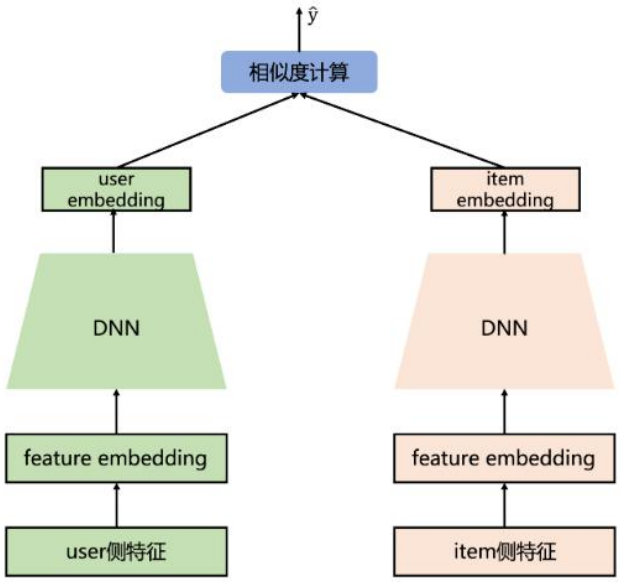
2.3 DSSM 双塔模型

参考： Learning Deep Structured Semantic Models for Web Search using Clickthrough Data

双塔模型（Two-Tower Model）是一种常用于推荐系统中的模型架构，用于对用户和物品进行建模，并计算它们之间的关联度。该模型的核心思想是将用户和物品分别表示为两个独立的塔（tower），然后通过度量它们之间的相似度或匹配程度来进行推荐。

在双塔模型中，用户塔和物品塔通常具有相似的结构，但参数是独立学习的。每个塔包含一系列层，例如嵌入层、隐藏层和输出层。这些层的设计可以根据具体任务和数据进行调整。

在推荐系统中，最为关键的问题是如何做好用户与 item 的匹配问题，因此对于推荐系统中 DSSM 模型的则是为 user 和 item 分别构建独立的子网络塔式结构，利用 user 和 item 的曝光或点击日期进行训练，最终得到 user 侧的 embedding 和 item 侧的 embedding。因此在推荐系统中，常见的模型结构如下所示：



3 基于预训练大模型的推荐技术

3.1.基于用户特征的推荐 - PeterRec（谈笑枫）

参考论文：

- Parameter-Efficient Transfer from Sequential Behaviors for User Modeling and Recommendation, SIGIR 2020
- UPRec: User-Aware Pre-training for Recommender Systems ，submitted TKDE in 2021
- U-BERT: Pre-training user representations for improved recommendation, AAAI 2021

- UserBERT: Self-supervised User Representation Learning , arxiv 2021
- One4all User Representation for Recommender Systems in E-commerce , arxiv 2021
- One Person, One Model, One World: Learning Continual User Representation without Forgetting, SIGIR 2021
- Scaling Law for Recommendation Models: Towards General-purpose User Representations , AAAI 2023

3.1.1 背景

新用户和冷用户喜好预测问题一直是推荐系统领域的一个难题，广泛存在于计算广告，App 推荐，电子商务和信息流推荐场景。目前绝大多数的解决方案都是基于用户外部画像数据进行喜好预测，因此预测准确率严重受制于画像数据准确率，并且用户画像数据搜集成本高，涉及敏感的隐私问题；另外，据笔者所知，即便具有十分精准的用户画像数据，新冷用户仍然很难做到个性化推荐，点击率和相应的 top-N 指标仍然显著低于常规热用户。 那么关于用户冷启动的场景，有没有其他更好的解决办法呢？最近，一篇腾讯 QQ 看点 (PCG 事业群) 团队 SIGIR2020 长文 Parameter-Efficient Transfer from Sequential Behaviors for User Modeling and Recommendation 提出了一种迁移学习架构 PeterRec 专门解决新用户和冷用户推荐。 PeterRec 基本思想是通过自监督学习一个通用的用户表征，然后将该用户表征应用到下游任务中，例如冷启动用户场景（PeterRec 同时可以解决用户画像预测）。从论文中的实验结果来看，这种采用自监督预训练网络学习用户点击行为的方法可以高效地推测出用户的偏好等信息。

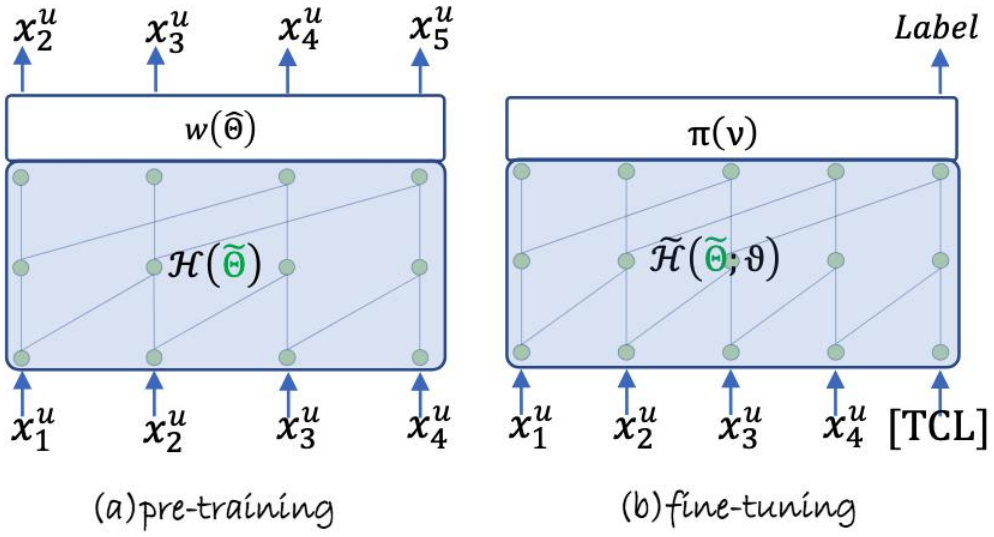
近年来，迁移学习对 CV 和 NLP 领域产生了重大的影响，但尚未被广泛应用于推荐系统领域，推荐系统领域目前相应的迁移学习科研工作都没有明确的展示出 pretrain 网络对于下游任务具有 positive transfer 效果。而在腾讯，由于具有非常丰富的业务场景，部分业务，例如腾讯视频，QQ 浏览器具有数亿的 DAU 用户，并且很多用户具有数百上千的点击行为，这些海量的用户点击行为为其他推荐业务场景（例如 QQ 看点，微视，腾讯广告， 应用宝，微信看一看）新冷用户提

供了丰富的可迁移的知识，QQ 团队尝试将 PeterRec 模型应用在 QQ 看点的视频推荐业务中。选择 PeterRec 模型除了其较好的个性化推荐能力外，很重要一点，PeterRec 可以实现一个 pretrain 网络服务数十/百个推荐业务场景能力。本文将从模型架构，数据处理，模型实现，后续工作这四个方面来介绍。

3.1.2 模型架构

首先 PeterRec 根据预训练的自监督方式可以分为单向自回归方式（autoregressive）和双向遮掩式，这一点类似于近期的 NLP 工作，例如 GPT。根据微调阶段模型补丁嫁接插入方式又可以分为串行插入（serial）和并行插入（parallel）。这里只介绍 autoregressive、serial 版本的 PeterRec 模型。

3.1.2.1 模型框架的输入输出



模型结构

Pretrain 阶段：

Pretrain 采用单向自回归的训练方式，根据用户观看的前 k 个视频预测其可能会看的下一个视频。输入是用户在腾讯视频看过的视频 id 序列[x1, x2, x3, · · · · · · , xn-1]，然后通过 embedding lookup 的方式获取每一个视频的隐

向量输入到预训练网络中；输出是对应的下一个视频 id，即[x2, x3,・・・・・・, xn-1, xn]。可以看到，PeterRec 模型不需要借助于任何图像和文本特征，仅需要用户点击视频的 ID 即可，视频的向量表示完全由模型训练得到，省去了特征工程的步骤，这种 pretrain 方式已经被应用于 CV 和 NLP 领域，并且取得了非常认可的效果，然而并没有在推荐系统领域得到推广。

Finetune 阶段：

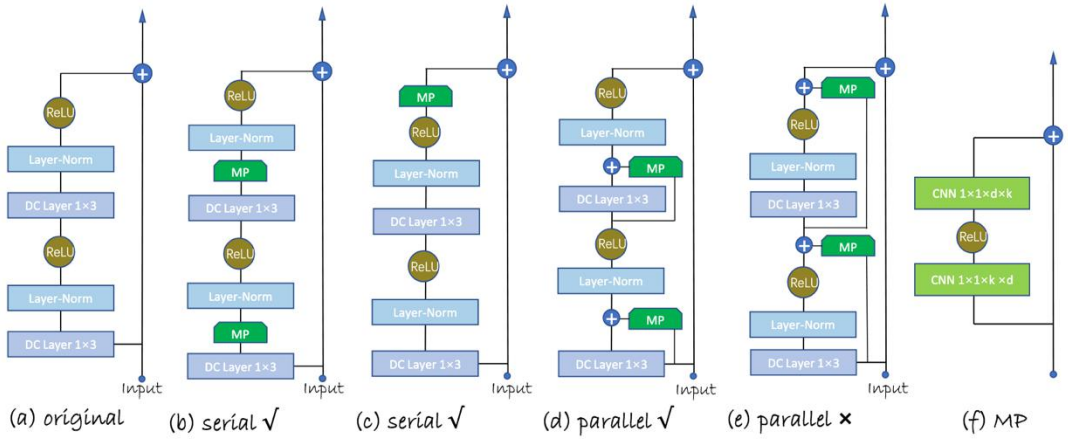
Finetune 阶段是根据用户在腾讯视频观看记录，预测其可能会在 QQ 看点感兴趣的视频。输入是[x1, x2, x3,・・・・・・, xn-1, xn, [CLS]]，　其中[x1, x2, x3,・・・・・・, xn-1, xn]用户在腾讯视频看过的视频 ID 序列, [CLS]是一个特殊的记号，表示在这个位置输出分类结果；输出 Label 是 QQ 看点的视频 ID，即预测用户在 QQ 看点可能会看的 top-N 个视频 ID。

3.1.2.2 模型的网络结构

Pretrain 阶段：

在模型 Pretrain 阶段，将其看成一个超大多分类问题。输入的视频 ID 序列经 embedding_lookup 操作后，提供给后面的空洞卷积网络。整个空洞卷积网络由若干个 residual block （如下图(a)所示）堆叠构成。在每个 block 中，包含两个空洞卷积层 （DC layer），每层的空洞因子以 2^n 增加。最后通过一个 softmax 层，预测出下一个视频。

相较于其他时序模型如 RNN、Transformer 等，PeterRec 模型基于空洞卷积神经网络构建大规模的预训练模型, 同时通过叠加空洞卷积层达到可视域指数级的增加，这种网络结构使得它在对超长的用户点击序列进行建模时更加高效。而相比之下，RNN 模型在对超长序列建模时，通常会遇到梯度消失和梯度爆炸的问题；而像 Transformer 这类 self-attention based 的模型，时间复杂度和显存需求会随着序列长度以二次方的级别增加。



residual block 和模型补丁

Finetune 阶段：

为了实现对预训练网络参数的最大化共享，Finetune 阶段仅对预训练模型做了两处的改动：

第一、在 residual block 中以串行的方式插入模型补丁（如上图(b)所示），每个模型补丁（如上图(f)所示）由一个瓶颈结构的残差块构成，且参数量仅有原始空洞卷积的不到十分之一。

第二、直接移除预训练 softmax 层，然后添加新任务的分类层。

微调通常要重新训练整个网络，并更新模型所有的参数，因此从参数量的角度来看，微调是非常低效的。相比这种微调所有参数的方式，PeterRec 模型在 finetune 阶段仅对模型补丁和新任务的 softmax 层中的参数进行更新，参数量大大减小的同时却可以达到与微调所有参数相当甚至更好的效果。而且，由于仅有少数参数参与更新，PeterRec 模型还具有很好的抗过拟合能力。

3.1.2.3 损失函数

Pretrain 阶段：

Pretrain 时采用 softmax 的多分类交叉熵损失函数。实际操作中，腾讯视频中

的视频经过各种过滤 ID 映射后还有数百万级别有效视频。如果采用 full softmax，训练效率会很低，所以这里采用了 tf.nn.sampled_softmax_loss，实际只采样了 20%的 item 作为负样本用做训练，当然其他 efficent 采样和 loss 设置也同样适用，例如 NCE loss 或者下文讲到的 LambdaFM 方式。

Finetune 阶段：

对于排序场景，pairwise 类方法要比 pointwise 类方法（直接看做分类或者回归）更合适，所以 finetune 采用了 pairwise ranking loss (BPR)。pairwise loss 里构造样本时需要同时考虑两个 item 比如 xi 和 xj ，这两个 item 是有顺序的，比如用户在排序列表里点击了 xi，而未点击 xj，可以看做 xi 要优于 xj。

因此，需要为每一个真实物品 label（y）采样一个负样本 y-，通过计算用户的隐向量与 y 和 y-的隐向量的内积作为两个 item 的打分 oy 和 oy-，然后算出最终的 BPR loss 如下：

$$R_{BPR}(\mathcal{T};\tilde{\Theta};v;\vartheta) = -\sum_{(u,y)\in\mathcal{T}} \log \delta(o_y - o_{y-})$$

3.1.3 数据处理

3.1.3.1 item 过滤与编码

过滤：

由于涉及到不同业务数据，腾讯视频业务流水均需要封闭域中安全获取，得到用户的原始观看序列后，需要过滤一些过热或过冷的视频 item（过热的视频没有区分度，无法看出用户特定的偏好；过冷的视频由于出现次数少，模型学得的隐向量很难准确反映视频的信息，并且没有充分的训练，很容易成为噪声而影响最终效果）。过滤后，腾讯视频的视频数量在 200w+的级别。

编码：

在 pretrain 阶段，采用了 sampled_softmax_loss 来代替 full softmax loss，tensorflow 的 sampled_softmax_loss 函数在进行负采样的时候，是通过 log_uniform_candidate_sampler 进行的，使用这个 sampler 时的效果会是：item 编号越小，它被采样为负样本的概率越大。针对这种情况，在对视频 item 进行编号时，是按照 item 在播放序列中的出现次数降序排列，然后从 0 开始编号。（原因可见下面`LambdaFM 负采样`）在 finetune 阶段，采用了 BPR loss，没有用到 log_uniform_candidate_sampler，因此可以不用按 item 频率进行编号。

3.1.3.2 样本构造

Pretrain 阶段：

首先从腾讯视频的流水数据中拿到用户 a 的播放序列。经过过滤（根据视频的播放时长和完播率，过滤掉自动播放的视频）和去重（对原始播放序列的相邻 item 去重）后， 取用户 a 最新的 50 个播放视频作为一个 pretrain 的训练样本[x1, x2, x3,・・・・・・, x50]（若用户的播放序列长度不足 50，则在前面填充[PAD]）。用户播放序列行为可以根据计算资源设置，如果具有较充足的计算资源，可以设置行为序列为更大，例如 200 甚至 1000。

Finetune 阶段：

以相同方式从 QQ 看点的流水数据中拿到用户 a 的播放行为[y1 y2, y3,・・・・・・, ym]。 这时，根据用户 a 在腾讯视频和 QQ 看点的播放序列，为用户 a 构造 m 条 finetune 的训练样本：[x1, x2, x3,・・・・・・, x50, [CLS], y1], [x1, x2, x3,・・・・・・, x50, [CLS], y2], ... , [x1, x2, x3,・・・・・・, x50, [CLS], ym]。（需要注意的是，只有腾讯视频和 QQ 看点的交集用户才能用于构造 finetune 的训练样本，预测时候则不需要。）

3.1.3.3 样本选择

经过上述的处理，对于不同观看历史的用户，PeterRec 模型预测出来的 top-N 结果已经具有一定的相关性。实际的 case 分析显示，这些强相关的视频还是容易出现得分较低于高频 item 情况，排在 top100 之外，但是在头部都出现了 item vocab 中最热的那些视频，由此可见高频 item 对模型的影响还是很大。为了缓解 top-N 推荐结果中的头部效应问题，减少高频 item 对模型的影响，经过尝试了不同的均衡正负样本的策略，其中下列两种较为有效：

高频降采样：

Word2vec 的实现中，会指定一个概率 P(wi)对高频词进行打压，同时保留所有的低频词。实际源码中，高频词在每个样本中被保留的概率实现如下：

```
// Calculate the probability of keeping 'word'.
real ran = (sqrt(vocab[word].cn / (sample * train_words)) + 1) * (sample * train_words) / vocab[word].cn;
```

其中，参数 sample 用于控制降采样的程度，sample 值越小，降采样强度越大，实际使用中需要根据 item 的频率分布来确定，一般取 0.001 ~ 0.00001。

于是，在构造 finetune 训练样本的时候，需要先根据概率分布对用户在 QQ 看点的播放序列[y1 y2, y3,・・・・・・, ym]进行一次降采样，按照一定比例丢弃一些高频的视频 item。然后再与其在腾讯视频的播放序列做拼接，得到 finetune 的训练样本。

LambdaFM 负采样：

关于负采样，常用的做法有两种（1）采用曝光未点击作为负样本；（2）从总的候选池子中随机取样。得到的（1）方法效果较差，因此采用从候选 item 池子随机选择 itemID，但这种方式仍然存在一定的缺陷，它采样出来的样本多数集中在长尾处，LambdaFM 论文中是这么描述的：

`In fact, it has been recognized that item popularity distribution in most real-world recommendation datasets has a heavy tail, following an approximate power-law or

exponential distribution. Accordingly, most non-positive items drawn by uniform sampling are unpopular due to the long tail distribution, and thus contribute less to the desired loss function. Based on the analysis, it is reasonable to present a popularity-aware sampler to replace the uniform one before performing each SGD.`

这里 *popularity-aware sampler* 的意思是让更受欢迎的 item 有更大的几率被采样为负样本，这个其实是符合直觉的，因为相比那些不受欢迎且用户没有观看的视频，那些受欢迎的但是用户没有观看的视频更具信息量，更能帮助发现用户的偏好。LambdaFM 论文提供了 3 中负采样方法，该模型采用第一种负采样方式。

finetune 训练过程中，采用的是 LambdaFM 中的 Static & Context-independent Sampler 进行负采样，即视频 j 被采样为负样本的概率 pj 与它的热度排名 rank(j) 呈正相关：

$$p_j \propto \exp\left(-\frac{\text{rank}(j)+1}{|I| * \rho}\right)$$

其中，rank(j)表示视频 j 在所有视频 item 集合中的热度排名， 表示阈值，通常取 0.3-0.5。

3.1.3.4 Predict 处理

前面的处理过程已经可以很精准的实现看点视频推荐的个性化或者相关性，随机挑选了两个实例见表 1。 同时，还能有效处理用户在腾讯视频和 QQ 看点的偏好不一致的问题：在具体的 case 分析中，可以发现部分用户在腾讯视频看的大多是卡通类的视频，而在 QQ 看点很少看这类视频，此时，PeterRec 模型推出的视频是更接近于他们在 QQ 看点看过的真实视频的，而非卡通类的视频。 主要原因可能是该部分用户在腾讯视频大多是学龄儿童甚至是学龄前期，主要是使用其父母账号观看腾讯视频，而在看点账号大多是其父母在使用，因此推出的视

频偏向于年轻父母偏好。

后续为了增加推荐 topN 结果中的多样性，在 Predict 的过程中做了一些改变：

Predict 处理

如上图所示，在为用户生成推荐列表时，不再是直接将用户在腾讯视频的播放序列输入到模型中，而是将其拆分成了两类子序列：

第一类是播放序列里的有效播放序列的子串。取子串的原因是，用户观看的相邻视频之间兴趣点比较一致，也就是用户会在某个时刻连续观看一些同类的视频。这使得子串里的视频大多属于同一类，用户的兴趣点明确，更有利于模型找到用户的偏好。

第二类是从用户的有效播序列中随机采样一些 item 来构造子序列，原因是用户的播放序列中往往包含了多个种类的视频，随机构造子序列引入了随机性，可以更好地丰富 topN 结果中包含的视频种类。

最后将用户对应的所有子序列的 top-N 结果，经过 concat、shuffle 和去重，得到用户最终的 top-N 推荐列表。

3.2 顺序/基于会话的推荐 （谈笑枫）

参考论文：

- BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer, CIKM 2019
- S3-Rec: Self-Supervised Learning for Sequential Recommendation with Mutual Information Maximization , CIKM-2020
- Transformers4Rec: Bridging the Gap between NLP and Sequential / Session-Based Recommendation, Recsys 2021
- Towards Universal Sequence Representation Learning for Recommender Systems , KDD 2022
- Learning Vector-Quantized Item Representation for Transferable Sequential Recommenders, WWW 2023

3.2.1 背景

传统的推荐系统算法通常使用单向的序列神经网络来对用户历史行为进行建模，这种方法只能利用单向的信息，存在一些限制和挑战。为了克服这些问题，提出了一种称为 BERT4Rec 的序列推荐模型，它采用了深层的双向自注意力机制来对用户行为序列进行建模。

BERT4Rec 模型通过 Cloze 目标进行序列推荐，即通过预测序列中随机遮蔽的项目（masked item），来学习双向表示模型。这种方法允许每个项目融合其左右两侧的信息，从而提高对用户行为序列的建模能力。此外，采用双向自注意力机制还可以避免信息泄漏，并有效地训练双向模型。

通过在四个基准数据集上进行大量实验，BERT4Rec 模型始终表现优于其他最新的序列模型。这表明采用双向自注意力机制和 Cloze 目标的序列推荐模型在建模用户动态偏好方面具有重要的优势。

BERT4Rec 是一种使用深层的双向自注意力机制进行序列推荐的模型。相比传统的单向模型，它可以更好地建模用户的历史行为，并在推荐系统中取得更好的性能。

精确地捕捉用户的兴趣是推荐系统的核心问题。在实际应用中，用户的兴趣是动态变化的，受到其历史行为的影响。例如，尽管在正常情况下不会购买游戏机配件，但在购买 Nintendo Switch 之后，用户可能会迅速购买配件，例如 Joy-Con 控制器。

为了捕捉用户偏好的动态变化，已经提出了许多根据用户历史交互信息的序列推荐算法。最早的方法使用马尔科夫模型对用户序列进行建模，但其中一些方法的强假设破坏了推荐系统的准确性。近期，一些序列神经网络在序列推荐问题中取得了不错的效果。基本思想是将用户的历史序列从左到右编码为一个向量，然后基于该向量进行推荐。

尽管这些方法具有普适性和有效性，但从左到右的单向模型无法充分学习用户行为序列的最佳表示。这主要是因为单向模型限制了历史序列中每个项目隐藏表示的能力，每个项目只能编码来自前面项目的信息。另一个限制是，之前的单向模型最初是针对具有自然顺序的序列数据（如文本和时间序列数据）引入的。它们经常要求数据严格按照顺序排列，但对于现实应用中的用户行为来说，并不总是合适的。实际上，由于各种不可观察的外部因素，用户在历史交互中选择项目的顺序可能不遵循严格的顺序假设。在这种情况下，将双向上下文合并到用户行为序列建模中变得至关重要。

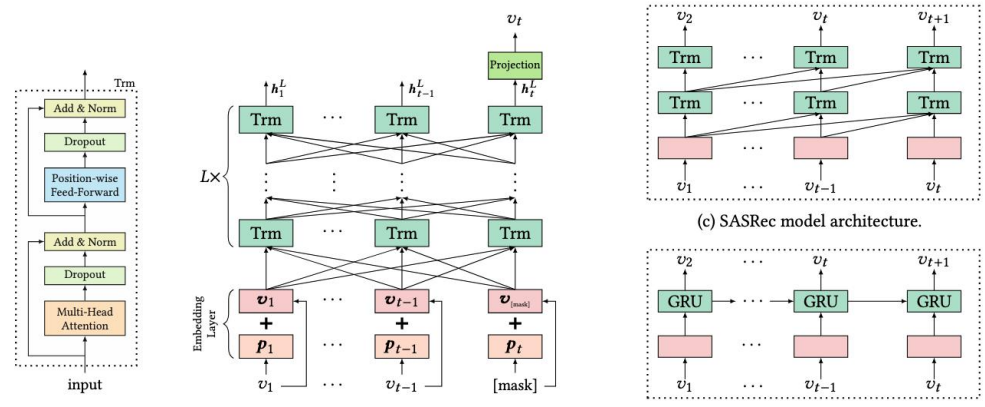
为了解决上述限制，本篇文章提出了一种基于双向自注意力和 Cloze 任务的用户行为序列建模方法。这是第一个将深度序列模型和 Cloze 任务引入推荐系统的研究。将它与最先进的方法进行了比较，并通过对四个基准数据集的定量分析，证明了本文算法的有效性。此外，对模型还进行了一项消融分析，分析了模型中关键组件的贡献。

综上所述，本篇文章提出了一种基于双向自注意力和 Cloze 任务的用户行为序列建模方法，旨在精确捕捉用户兴趣的动态变化。该方法在推荐系统中具有重要意义，并通过实验证明了其优越性。

3.2.2 模型

如下图（b）所示，含有 L 层的 Transformer，每一层利用前一层所有的信息。相比于图（d）基于 RNN 的推荐模型，self-attention 可以捕获任意位置的信息。相比于基于 CNN 的推荐模型，可以捕获整个 field 的信息。相比于图（c）和图（d）的模型（都是 left-to-right 的单向模型），本文提出的双向模型可以解决现有模型的问题。

Transformer 层 如图 (a) 所示, Transformer 由两部分组成 Multi-Head Self-Attention 和 Position-wise Feed-Forward network 部分。



(a) Transformer Layer. (b) BERT4Rec model architecture. (c) SASRec model architecture. (d) RNN based sequential recommendation methods.

Figure 1: Differences in sequential recommendation model architectures. BERT4Rec learns a bidirectional model via Cloze task, while SASRec and RNN based methods are all left-to-right unidirectional model which predict next item sequentially.

（1）Multi-Head Self-Attention 对于模型框架中的第 L 层 Transformer，输入为 H_l ,首先是 Multi-Head Self-Attention 过程：

其次是 Dropout 和 Add & Norm 过程，这里的 Add 就是 Skip Connection 操作，目的是反向传播时防止梯度消失的问题；而 Norm 是 Layer Norm 操作。

(2) Position-wise Feed-Forward Network 由于只有线性映射，为了使得模型具有非线性的性质，所以采用了 Position-wise Feed-Forward Network。Position-wise 的意思是说，每个位置上的向量分别输入到前向神经网络中，计算方式如下：

这里采用的激活函数是 Gaussian Error Linear Unit (GELU)，而非 RELU，其出自论文《Gaussian error linear units (gelus)》。GELU 在 RELU 的基础上加入了统计的特性，在论文中提到的多个深度学习任务中都取得了较好的实验结果。

(3) Embedding 层：在没有任何 RNN 或 CNN 模块的情况下，Transformer 不知道输入序列的顺序。为了利用输入的顺序信息，在 Transformer 的 Embedding 层加入了位置嵌入，本文的位置向量是学到的，不是 transformer 中的正弦。位置向量矩阵可以给定任意位置的向量，但是要明确最大的长度，因此需要对输入序列进行截断。

(4) Output 层 ： 这里使用两层带有 GELU 激活函数的前馈网络得到最终的输出。

BERT4Rec 模型在输入层和输出层用了共享的物品 embedding 矩阵，目的是减轻过拟合和减少模型大小。

模型训练和预测的目的是预测用户下一个要交互的物品 v+1,对于传统的序列推荐模型，如上图（d）中的 RNN 模型，输入是[v0, v1, ...vt]，转换为对应的输出为[v1, ...vt+1]，那么自然可以拿最后一个时刻输出的物品进行推荐。而在 BERT4Rec 中，由于是双向模型，每一个 item 的最终输出表示都包含了要预测物品的信息，这样就造成了一定程度的信息泄漏。因此采用 Cloze taske，也就是将输入序列中的 p%的物品进行 masked，然后根据上下文信息预测 masked 的物品。

在训练阶段，为了提升模型的泛化能力，让模型训练到更多的东西，同时也能够创造更多的样本，借鉴了 BERT 中的 Masked Language Model 的训练方式，随机的把输入序列的一部分掩盖（即变为[mask]标记），让模型来预测这部分盖住地方对应的物品：

这里的 masked，跟 bert 的 LM 一样，提升泛化能力。

采用这种训练方式，最终的损失函数为：

如上所述，在训练过程和最终的序列预测推荐任务之间是不匹配的。因为 Cloze task 的目的是预测当前被 masked 的物品，而序列预测推荐的目的是预测未来。为了解决这个问题，在预测阶段将 masked 附加到用户行为序列的末尾，然后根据该 masked 的最终隐藏表示来预测下一项。

为了更好地匹配序列推荐任务(即，预测最后一项)，在训练过程中还生成了只 mask 输入序列中最后一项的样本。这个工作就像对序列推荐的微调一样，可以进一步提高推荐性能。

3.2.3 实验

首先，论文对比了 BERT4Rec 模型和一些 Base 模型在 4 个数据集上的表现，发现 BERT4Rec 模型相比于 Base 模型，其性能都有较大的提升。

其次，是对 Embedding 的长度 、训练时 mask 物品的比例和序列的最大长度等参数的对比。结论为：Embedding 长度越长，模型的效果更好；对于不同的数据集，最佳 mask 的比例并不相同；对于不同的训练集，最佳的序列长度也不相同。

最后，是对模型结构的一些对比实验，主要有是否使用 PE(positional embedding)，是否使用 PFFN(position-wise feed-forward network)，是否使用 LN(layer normalization)，是否使用 RC (即 Add 操作，residual connection)，是否使用 Dropout，以及 Transformer Layer 的层数和 Multi-head Attention 中 head 的个数。

3.2.4 结论

总之，BERT4Rec 就是把 BERT 用在推荐系统中，知道用户的播放（购买、点击、...)序列 item1, item2, item3，预测下一个播放的 item 问题。训练的时候使

用 Mask LM 任务使用海量用户行为序列进行训练，模型评估时将序列的最后一个 item 进行 masked，预测的时候在序列的最后插入一个“[mask]”，然后用“[mask]”增强后的 embedding 预测用户接下来会观看哪个 item。整体来说，该篇模型为 BERT 在推荐系统领域的工业界落地提供了强有力的指导说明，但在推荐系统领域的学术界来说创新性就显得不是很大。

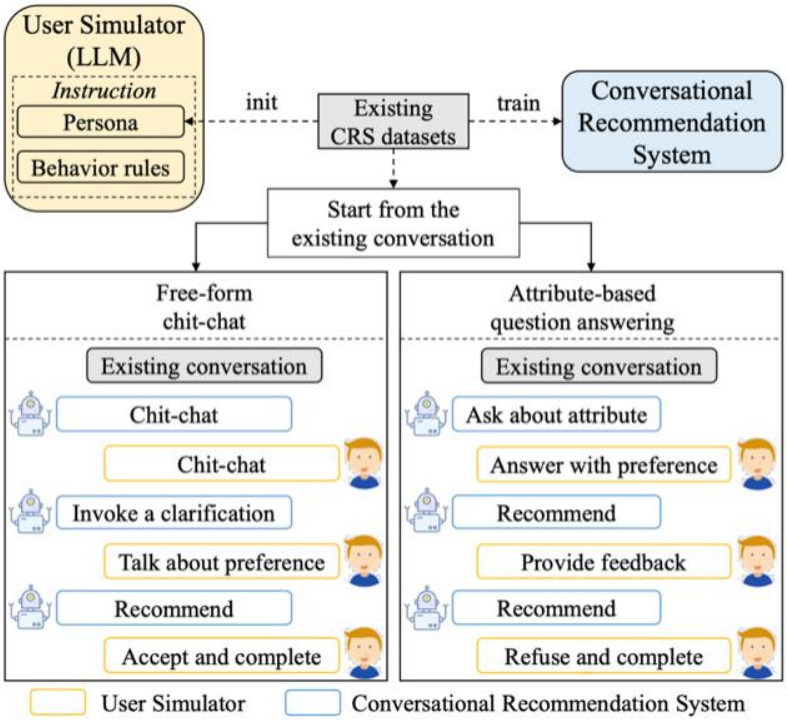
BERT 的训练和预测耗时耗资源，如何提高 BERT 的在线服务能力是一个业界需要考虑的问题。其次，论文中没有使用物品、用户属性和场景的信息，只使用了行为信息，如何把额外的信息加入模型也是值得重点探索的。

3.3 基于预训练后的大语言模型进行推荐

3.3.1 基于 ChatGPT 来构建对话式推荐系统（张宏伟）

参考 Rethinking the Evaluation for Conversational Recommendation in the Era of Large Language Models

利用 ChatGPT 来构建对话式推荐系统，并为此进行了系统性的评测。



文章首先在已有的 benchmark 数据集上评测了 ChatGPT 的对话推荐能力。发现 ChatGPT 并没有展示较好的效果。为此，作者检查了 ChatGPT 失败的案例，并发现失败的原因在于：已有的评测方式依赖于对齐人类手工标注的推荐和对话，并过分强调了基于对话上下文来对 ground-truth 物品的拟合。因此，传统的评测指标，如 BLEU 和 GOUGE 等无法反映 LLM 在文本生成任务上的真实能力。

尝试使用基于 LLM 的用户模拟器来测试 LLM 的对话推荐能力。在这样的评测方式下，ChatGPT 取得了出色的表现。特别的，ChatGPT 具有突出的解释能力，这是目前的对话推荐系统难以做到的。

文章提出了一个新的评估方法 iEvaLM，能够更好地反映 CRSs 系统的互动性质；展示了对于交互推荐任务而言，ChatGPT 具有比其他模型更为优秀的解释性能表现。文章结论：1）建议开发新的评估协议，考虑推荐系统的解释性和用户体验等因素；2）验证 ChatGPT 模型在解释性能方面的优势，推荐引擎需要发展更复杂的解释能力。

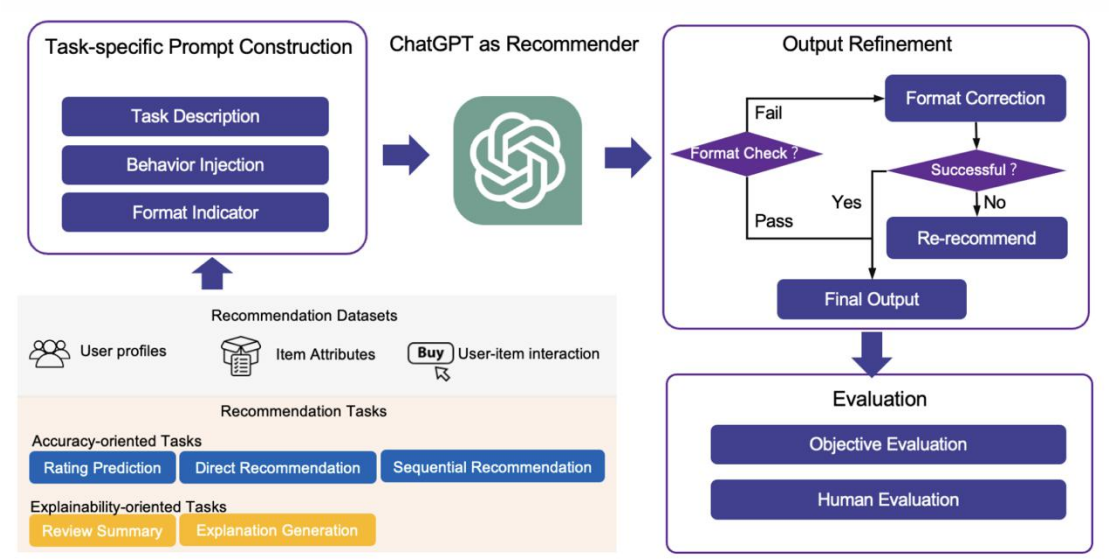
3.3.2 探索 ChatGPT 作为通用推荐模型的能力 （谈笑枫）

参考 Is ChatGPT a Good Recommender? A Preliminary Study

该文章设计了一系列的 prompt 并评估了 ChatGPT 在五种推荐场景的性能，包括评级预测，序列推荐、直接推荐、解释生成和评论生成。 与传统的推荐方法不同，该文章不会在整个过程中对 ChatGPT 进行微调，仅依靠 prompt 本身将推荐任务转化为自然语言任务。 此外，探索利用 few-shot 提示注入包含用户潜在兴趣的交互信息，帮助 ChatGPT 更好了解用户的需求和兴趣。

3.3.1 实验方法

具体的方法模型流程图如下所示：

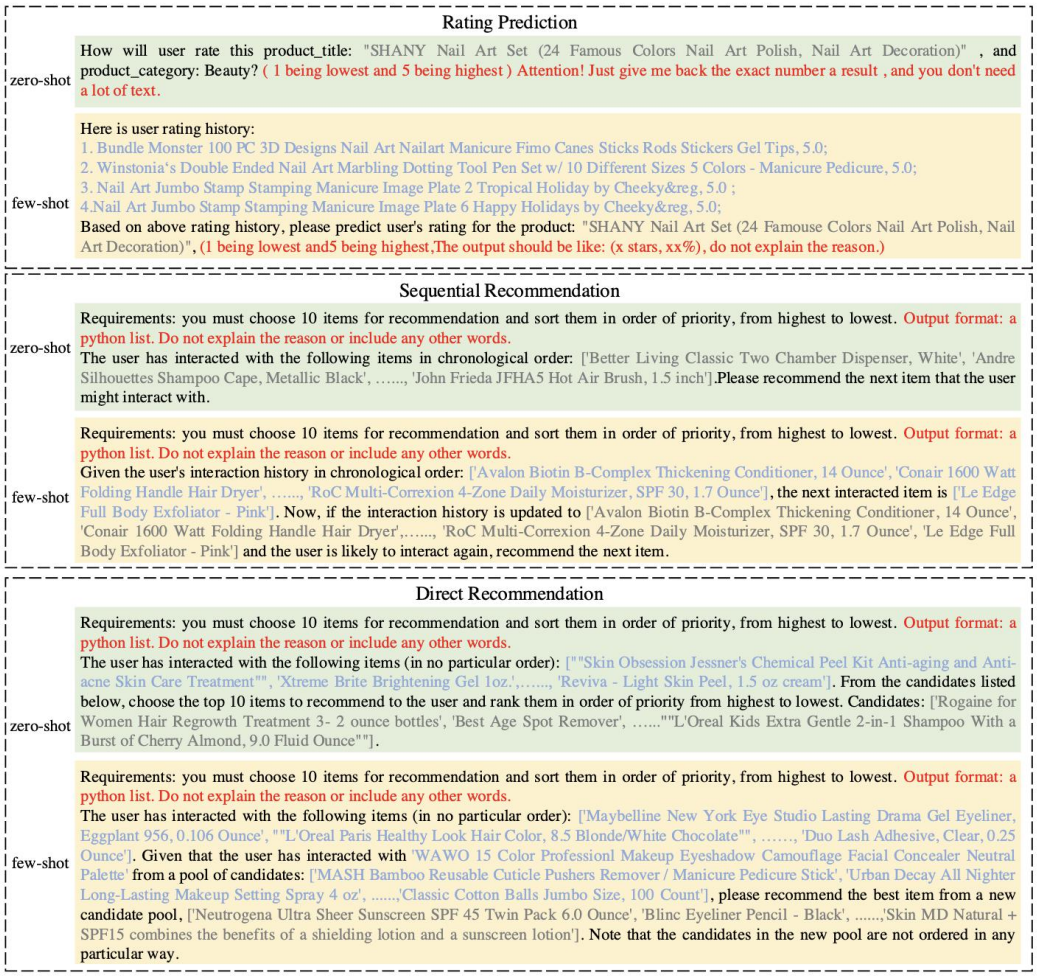


chatgpt 应用于推荐的流程图

如图所示，文章提出的方法主要包括两个步骤，任务特定的 prompt 构造和输出细化。

1.任务特定的 prompt 构造：

文章分别给出在 zero-shot 和 few-shot 下各种任务的 prompt 构造的样例，如下所示：



2. 输出细化

为了确保生成结果的多样性，ChatGPT 在其响应生成过程中加入了一定程度的随机性，这可能会导致对相同输入的不同响应。 然而，当使用 ChatGPT 进行推荐时，这种随机性有时会导致评估推荐 item 的困难。 而 prompt 构造中的格式指示符可以部分缓解了这个问题，但在实际使用中，仍然不能保证预期的输出格式。 因此他们设计输出细化模块检查 ChatGPT 输出的格式。 如果输出通过格式检查，直接作为最终输出。 如果不是，则根据预定义的规则对其进行修改。 如果格式修正成功，修正后的结果作为最终输出。 如果没有，则将相应的提示输入 ChatGPT 重新推荐，直到满足格式要求。 值得注意的是不同的任务有不同的输出格式评估 ChatGPT 时的要求。 例如，对于评分预测，只需要一个特定的分数，而对于序列或直接推荐，需要推荐 item 列表。特别是对于序列推荐，一次性将数据集中的所有 item 发送到 ChatGPT 是一个挑战。 因此，ChatGPT 的输

出可能无法正确匹配数据集中的 item 集。 到针对这个问题，我们引入了一种基于文本匹配的方法校正过程中映射 ChatGPT 预测的相似性回到原始数据集。这种方式虽然不能完美体现 ChatGPT 的能力，但还是可以间接展示它在序列推荐中的潜力。

3.3.2 实验结果

Table 5: Performance comparison on review summarization (%).

Methods	Beauty			
	BLUE4	ROUGE1	ROUGE2	ROUGEL
T0	1.2871	1.2750	0.3904	0.9592
GPT-2	0.5879	3.3844	0.6756	1.3956
P5-B	**2.1225**	**8.4205**	**1.6676**	**7.5476**
ChatGPT(zero-shot)	0.0000	3.8246	0.2857	3.1344
ChatGPT(few-shot)	0.0000	2.7822	0.0000	2.4328

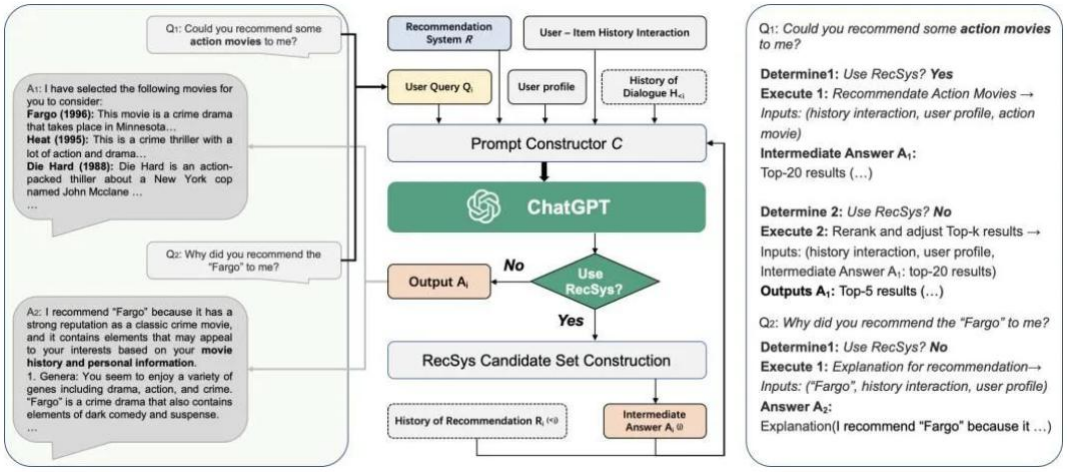
通过评估ChatGPT在推荐任务中的表现，并将其与传统推荐模型进行了比较。实验结果显示，ChatGPT 在评分预测方面表现良好，但在顺序和直接推荐任务中表现较差，这表明有必要进一步探索和改进这些方面。尽管存在一些限制，但 ChatGPT 在可解释性推荐任务的人工评估中优于现有方法，突显了其在生成解释和摘要方面的潜力。项研究为了解 ChatGPT 在推荐系统中的优势和局限性提供了宝贵的见解，有可能能激发未来的研究探索如何利用大型语言模型提升推荐性能。

3.3.3 用 LLMs 增强传统推荐的范式（张宏伟）

推荐系统已被广泛部署用于自动推断人们的偏好并提供高质量的推荐服务。然而大多数现有的推荐系统仍面临诸多缺陷，例如缺少交互性、可解释性，缺乏反馈机制，以及冷启动和跨域推荐。

文章提出了一种用 LLMs 增强传统推荐的范式 Chat-Rec（ChatGPT Augmented Recommender System）。通过将用户画像和历史交互转换为 Prompt，Chat-Rec 可以有效地学习用户的偏好，它不需要训练，而是完全依赖于上下文学习，并可以有效推理出用户和产品之间之间的联系。通过 LLM 的增强，在每次对话后都可以迭代用户偏好，更新候选推荐结果。

3.4.3.1 框架



如图所示，Chat-Rec 将用户与物品的历史交互、用户档案、用户查询 和对话历史 （如果有的话）作为输入，并与任何推荐系统 R 接口。如果任务被确定为推荐任务，该模块使用 R 来生成一个候选项目集。否则，它直接向用户输出一个响应，如对生成任务的解释或对项目细节的要求。提示器模块需要多个输入来生成一个自然语言段落，以捕捉用户的查询和推荐信息。这些输入如下：

1. 用户与物品的历史交互，指的是用户过去与物品的互动，比如他们点击过的物品，购买过的物品，或者评价过的物品。这些信息被用来了解用户的偏好并进行个性化推荐。

2. 用户画像，其中包含关于用户的人口统计和偏好信息。这可能包括年龄、性别、地点和兴趣。用户资料有助于系统了解用户的特点和偏好。

3. 用户查询 ，这是用户对信息或建议的具体要求。这可能包括他们感兴趣的一个具体项目或流派，或者是对某一特定类别的推荐的更一般的请求。

3.4.3.2 基于候选集压缩的推荐

传统的推荐系统通常会生成少量经过排序的候选产品，每个产品都有一个反映系统推荐信心或结果质量的分数。然而，但由于产品集规模巨大，现有系统的性能仍有很大改进空间。

文章提出了一种使用 LLMs 的方法，通过缩小候选集的范围来提高推荐系统的性能。通过 LLMs，我们将用户的资料和历史交互转换为 Prompt。一旦 LLMs 了解了用户的偏好，推荐系统生成的候选集就会提供给 LLMs。LLMs 可以根据用户的偏好进一步过滤和排序候选集。这种方法可以确保向用户展示一个更小、更相关的物品集，增加他们找到自己喜欢的东西的可能性。

3.4.3.3 实验

实验中使用的数据集是 MovieLens 100K，随机筛选了 200 个用户组成数据集

Top-k 推荐任务

实验表明，Chat-Rec 可以根据用户偏好进一步优化推荐系统推荐的候选集。Chat-Rec 框架在 NDCG 和 Precision 指标上都能超过 LightGCN，其中 davinci-003 的效果最好。

Chat-Rec 最重要的能力是优化推荐系统的候选集，将用户可能喜欢但在推荐系统的候选集中被放在后面的电影排到前面。这需要应用 LLMs 的电影知识，理解用户的偏好，以及两者之间的匹配关系的能力。

为了证实这一发现，在同一对话中再次询问 LLMs 关于那些出现在推荐系统前 5 名但没有出现在 LLMs 前 5 名的电影。LLMs 的反馈表示，用户不太可能会喜欢这部电影，或者很难确定用户是否会喜欢这部电影，并给出了明确的理由。不一致的情况显示，Chat-Rec 的建议是完全基于对用户偏好和电影信息的理解。

Models	Precision	Recall	NDCG
LightFM	0.2830	0.1410	0.2846
LightGCN	0.3030	**0.1455**	0.3425
Chat-Rec (gpt-3.5-turbo)	0.3103	0.1279	0.3696
Chat-Rec (text-davinci-003)	**0.3240** (+6.93%)	0.1404 (-3.51%)	**0.3802** (+11.01%)
Chat-Rec (text-davinci-002)	0.3031	0.1240	0.3629

zero-shot 评分预测

LLMs 可以有效地从用户画像和历史互动中学习用户偏好，而不需要任何的训练，就可以准确预测用户对候选电影的评分。text-davinci-003 取得了最好的结果

Models	RMSE	MAE
MF	0.988	0.771
Item-KNN	0.933	0.734
CHAT-REC (gpt-3.5-turbo)	0.969	0.756
CHAT-REC (text-davinci-003)	**0.785**	**0.593**
CHAT-REC (text-davinci-002)	0.8309	0.6215

3.4.3.3 总结

LLM 可以基于 in-context learning 较好的理解用户偏好以及构建用户和物品之间的联系。因此，LLM 在向用户推荐合适物品的同时，也可以向用户提供个性化的解释结果

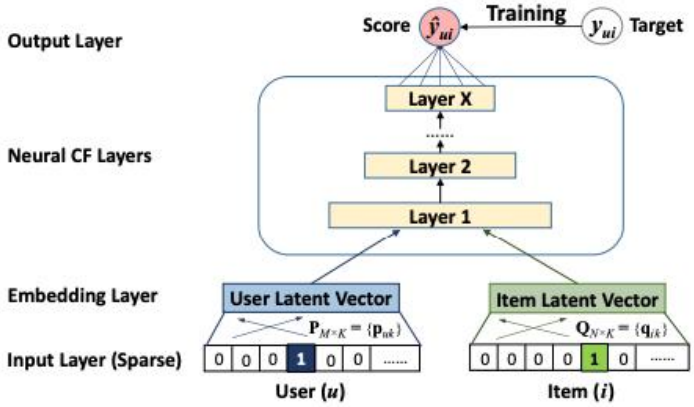
3.4 基于双塔模型进行推荐（张宏伟）

下面介绍基于双塔模型进行推荐的技术：

3.4.1 基于双塔模型进行协同过滤推荐

参考 "Neural Collaborative Filtering" by Xiangnan He, et al. (WWW 2017)

在传统的协同过滤方法中，通常使用矩阵分解来捕捉用户和物品之间的潜在关系。然而，这种方法忽略了用户和物品特征之间的交互作用，限制了推荐系统的准确性和个性化能力。提出了神经协同过滤（NCF）模型，是**一种基于双塔模型的协同过滤推荐模型**。NCF 通过学习用户和物品的嵌入表示，并使用神经网络进行交互和预测，提高了推荐的准确性和个性化能力。

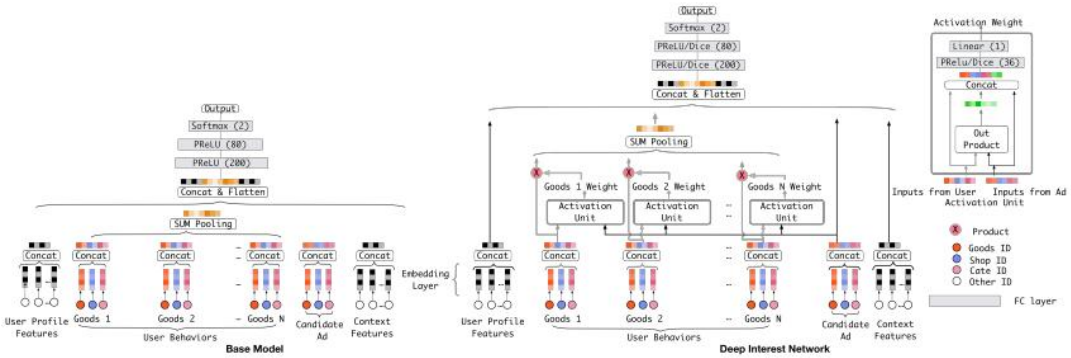


NCF 模型的核心思想是通过神经网络学习用户和物品之间的交互函数。与传统的矩阵分解方法不同，NCF 使用了多层感知器（Multi-Layer Perceptron，MLP）来建模用户和物品特征之间的非线性交互。通过引入非线性函数，NCF 能够更好地捕捉用户和物品之间的复杂关系，从而提高推荐的准确性和个性化能力。

3.4.2 基于双塔模型进行点击率预测推荐

参考 Deep Interest Network for Click-Through Rate Prediction" by Guorui Zhou, et al. (KDD 2018)

在在线广告推荐系统中，点击率预测是一个关键的问题。传统的点击率预测方法主要基于浅层模型，往往只考虑用户和物品之间的显性特征，而忽略了用户的兴趣偏好以及隐性的关联信息。为了解决这个问题，**DIN 模型是一种基于双塔模型的点击率预测推荐模型**，提出了一种新颖的兴趣抽取机制，通过动态地对用户的兴趣进行建模，提升了点击率预测的准确性和个性化能力。



DIN 模型的核心思想是利用注意力机制来对用户兴趣的不同部分进行加权。具体而言，DIN 模型首先使用一个嵌入层将用户和物品的特征转化为低维稠密向量表示。然后，通过计算用户的兴趣与每个物品的相似度得分，利用注意力机制

为不同物品赋予不同的权重。这样，DIN 模型能够更加关注用户感兴趣的物品，提高点击率预测的准确性。

此外，DIN 模型还引入了候选物品池和历史兴趣池的概念，用于更好地捕捉用户的历史行为和兴趣演化。通过将候选物品池和历史兴趣池的信息进行交互和融合，DIN 模型能够更好地利用用户的历史行为，进一步提高点击率预测的性能。

论文中的实验证明了 DIN 模型在点击率预测任务上的优越性。与传统的浅层模型相比，DIN 模型在准确性和个性化能力方面都取得了显著的改进。特别是在用户兴趣偏好较为复杂的情况下，DIN 模型能够更好地捕捉到用户的兴趣，提供更准确和个性化的推荐结果。

综上所述，《Deep Interest Network for Click-Through Rate Prediction》论文提出了一种基于注意力机制的深度学习模型，即 DIN 模型，用于点击率预测任务。该模型通过动态建模用户的兴趣和利用注意力机制来加权用户对不同物品的关注，提高了点击率预测的准确性和个性化能力。该论文的研究成果对广告推荐系统中点击率预测的深度学习方法具有重要的意义和应用价值。

3.4.3 基于双塔模型进行大规模广告预推荐

参考 IntTower: the Next Generation of Two-Tower Model for Pre-Ranking System, CIKM 2022

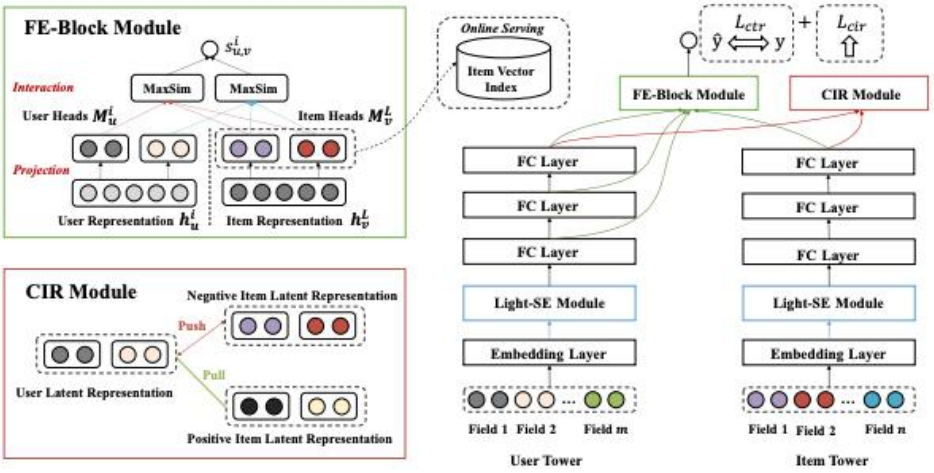
关于在大规模广告预推荐系统中改进预测准确性和维持推理效率的技术。以往的模型包括 LR，FM，DSSM 和 DAT，但难以平衡信息交互和推理效率。本文提出了 Interaction enhanced Two-Tower model (IntTower)，IntTower 旨在通过增强用户和物品塔之间的信息交互，同时保持高推理效率，提升双塔模型的性能。

IntTower 模型采用 Light-SE，FE-Block 和 CIR 模块，用于强化用户和物品特征的交互。

Light-SE 模块通过全局平均池化实现对特征内部信息的提取，加强特征交互，识别不同特征的重要性，并获得精细的特征表示。

FE-Block 模块通过低次数多项式空间模拟并提供细粒度交互模型。明确捕获用户和物品塔之间的交互信号。

CIR 模块通过对比损失函数对交互权重加权，增强模型的交互建模能力。



综合实验结果表明，IntTower 在面向预排系统时的效果明显优于传统的双塔模型，甚至超过了基于深度学习的排名模型 DeepFM。此外，对于大规模广告预排系统，IntTower 具有与排名模型相当的预测准确性和更高的推断效率。

4.相关代码尝试（张宏伟）

4.1 验证 ChatGPT 在推荐系统中的功能

相关代码来自 Uncovering ChatGPT’s Capabilities in Recommender Systems, arxiv 2023,

```
import time
import subprocess
from xpflow import Xp


class MyArgs(Xp):
    model = ["text-davinci-003"]
    domain = ["Movie"]
    task = ["list"]
    no_instruction = False
```

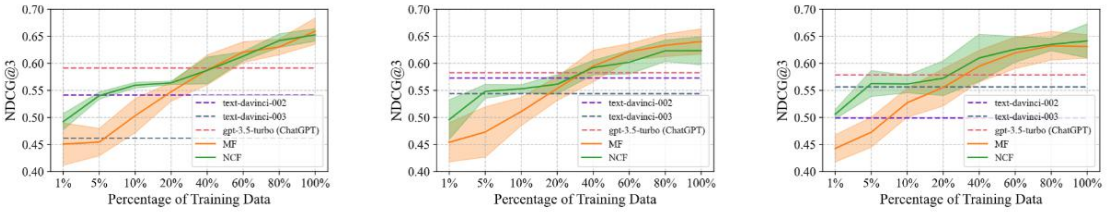
```
    example_num = [1]
    begin_index = 5
    end_index = 505
    api_key = "YOUR API-KEY"
    max_requests_per_minute = 2000
    max_tokens_per_minute = 80000
    max_attempts = 100
    proxy = None

if __name__ == "__main__":
    for i, args in enumerate(MyArgs()):
        cmd = (
            f"python src/main.py "
            f"--model {args.model} "
            f"--domain {args.domain} "
            f"--task {args.task} "
            f"--example_num {args.example_num} "
            f"--begin_index {args.begin_index} "
            f"--end_index {args.end_index} "
            f"--api_key {args.api_key} "
            f"--max_requests_per_minute {args.max_requests_per_minute} "
            f"--max_tokens_per_minute {args.max_tokens_per_minute} "
            f"--max_attempts {args.max_attempts} "
            f"--proxy {args.proxy} "
        )
        if args.no_instruction:
            cmd += "--no_instruction "
        start = time.time()
        print("-"*20)
        print(args)
        p = subprocess.Popen(cmd, shell=True)
        p.wait()
        end = time.time()
        print(f"Cmd: {i} finished! time cost(s): {end - start:.2f}")
        print("-"*20)
```

相应结果

Table 1. Overall performance of different models on four datasets from different domains. Bold indicates the best result for each row and '_' indicates the best result for each LLM. 'random' denotes recommendation based on a random policy. 'pop' denotes recommendation based on items' popularity judged by the number of interactions.

Domain	Metric	random	pop	text-davinci-002			text-davinci-003			gpt-3.5-turbo (ChatGPT)		
				point-wise	pair-wise	list-wise	point-wise	pair-wise	list-wise	point-wise	pair-wise	list-wise
Movie	Compliance Rate	-	-	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	99.98%	100.00%
	NDCG@1	0.2000	0.2240	0.3110	0.3203	0.2600	0.2259	0.2843	0.3260	**0.3342**	0.3230	0.3320
	NDCG@3	0.4262	0.4761	0.5416	0.5728	0.4990	0.4618	0.5441	0.5564	**0.5912**	0.5827	0.5785
	MRR@3	0.3667	0.4103	0.4824	0.5071	0.4363	0.3998	0.4763	0.4950	**0.5260**	0.5162	0.5167
Book	Compliance Rate	-	-	99.96%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	99.98%	99.80%
	NDCG@1	0.2000	0.2440	0.2420	0.2847	0.2000	0.2325	0.2887	0.2440	0.2823	0.3061	**0.3126**
	NDCG@3	0.4262	0.4999	0.4889	0.5298	0.4290	0.4585	0.5293	0.4597	0.5075	0.5350	**0.5395**
	MRR@3	0.3667	0.4340	0.4247	0.4646	0.3690	0.3993	0.4665	0.4040	0.4495	0.4774	**0.4800**
Music	Compliance Rate	-	-	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	99.96%	99.80%
	NDCG@1	0.2000	0.1780	0.2354	0.2397	0.2300	0.2377	0.2690	0.2540	0.2892	0.3077	**0.3086**
	NDCG@3	0.4262	0.4094	0.4623	0.4681	0.4277	0.4732	0.5072	0.4506	0.5201	0.5439	**0.5567**
	MRR@3	0.3667	0.3470	0.4030	0.4082	0.3750	0.4113	0.4448	0.4000	0.4605	0.4830	**0.4950**
News	Compliance Rate	-	-	99.80%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	99.60%
	NDCG@1	0.2000	**0.3080**	0.2183	0.2200	0.2920	0.2532	0.2630	0.2540	0.2591	0.2491	0.2892
	NDCG@3	0.4262	**0.5444**	0.4483	0.4550	0.5059	0.4880	0.4892	0.4742	0.4826	0.4991	0.5094
	MRR@3	0.3667	**0.4840**	0.3879	0.3936	0.4497	0.4271	0.4294	0.4173	0.4246	0.4354	0.4515



(a) point-wise (b) pair-wise (c) list-wise

4.2Bert 模型在推荐系统应用

代码来源：https://github.com/FeiSun/BERT4Rec
核心代码（bert 模型）

```
class BertModel(object):

  def __init__(self,
               config,
               is_training,
               input_ids,
               input_mask=None,
               token_type_ids=None,
               use_one_hot_embeddings=True,
               scope=None):

    ...

    with tf.variable_scope(scope, default_name="bert"):
      with tf.variable_scope("embeddings"):
        ...
        # Perform embedding lookup on the word ids.
```

```
            (self.embedding_output,
             self.embedding_table) = embedding_lookup(
                input_ids=input_ids,
                vocab_size=config.vocab_size,
                embedding_size=config.hidden_size,
                initializer_range=config.initializer_range,
                word_embedding_name="word_embeddings",
                use_one_hot_embeddings=use_one_hot_embeddings)

            # Add positional embeddings and token type embeddings, then layer
            # normalize and perform dropout.
            self.embedding_output = embedding_postprocessor(
                input_tensor=self.embedding_output,
                use_token_type=True,
                token_type_ids=token_type_ids,
                token_type_vocab_size=config.type_vocab_size,
                token_type_embedding_name="token_type_embeddings",
                use_position_embeddings=True,
                position_embedding_name="position_embeddings",
                initializer_range=config.initializer_range,
                max_position_embeddings=config.max_position_embeddings,
                dropout_prob=config.hidden_dropout_prob)

        with tf.variable_scope("encoder"):
            # This converts a 2D mask of shape [batch_size, seq_length] to a 3D
            # mask of shape [batch_size, seq_length, seq_length] which is used
            # for the attention scores.
            attention_mask = create_attention_mask_from_input_mask(
                input_ids, input_mask)

            # Run the stacked transformer.
            # `sequence_output` shape = [batch_size, seq_length, hidden_size].
            self.all_encoder_layers = transformer_model(
                input_tensor=self.embedding_output,
                attention_mask=attention_mask,
                hidden_size=config.hidden_size,
                num_hidden_layers=config.num_hidden_layers,
                num_attention_heads=config.num_attention_heads,
                intermediate_size=config.intermediate_size,
                intermediate_act_fn=get_activation(config.hidden_act),
                hidden_dropout_prob=config.hidden_dropout_prob,
                attention_probs_dropout_prob=config.
                attention_probs_dropout_prob,
                initializer_range=config.initializer_range,
                do_return_all_layers=True)

        self.sequence_output = self.all_encoder_layers[-1]

    def get_sequence_output(self):
```

```
    """Gets final hidden layer of encoder.

    Returns:
    float Tensor of shape [batch_size, seq_length, hidden_size] corresponding
    to the final hidden of the transformer encoder.
    """
    return self.sequence_output

def get_all_encoder_layers(self):
    return self.all_encoder_layers

def get_embedding_output(self):
    """Gets output of the embedding lookup (i.e., input to the transformer).

    Returns:
    float Tensor of shape [batch_size, seq_length, hidden_size] corresponding
    to the output of the embedding layer, after summing the word
    embeddings with the positional embeddings and the token type embeddings,
    then performing layer normalization. This is the input to the transformer.
    """
    return self.embedding_output

def get_embedding_table(self):
    return self.embedding_table
```

上述代码定义了 BertModel 类，该类实现了 BERT 模型的核心功能。在初始化过程中，它接受一些参数，如模型配置、训练标志、输入 ID、输入掩码、标记类型 ID 等，并构建了 BERT 模型的计算图。

- 在嵌入层中，通过 embedding_lookup 函数对输入 ID 进行嵌入查询，得到嵌入输出和嵌入表。然后，对嵌入输出进行位置嵌入和标记类型嵌入，并进行层归一化和 dropout 处理。
- 在编码器层中，根据输入掩码创建注意力掩码，并调用 transformer_model 函数运行堆叠的 Transformer 层。transformer_model 函数返回所有编码器层的输出，而 self.all_encoder_layers 保存了这些输出。
- get_sequence_output 方法返回最终的隐藏层输出，即 Transformer 编码器的最后一层的输出。
- get_all_encoder_layers 方法返回所有编码器层的输出。

- get_embedding_output 方法返回嵌入层的输出，即输入到 Transformer 的嵌入层输出。
- get_embedding_table 方法返回嵌入表。

5.小组分工：

每个成员在负责自己的任务的同时，也保持团队合作和协调，相互交流学到的知识。小组成员相互帮助、合作完成此次 survey 期末作业。

张宏伟（10205102417）：资料查询，论文撰写、整理修改，代码编程、调试， PPT 制作和汇报、参考文献整理

谈笑枫（10205102414）： 资料查询，论文撰写， PPT 制作和汇报

参考文献

【1】Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT) (pp. 4171-4186).

【2】Xiao, C., Xie, R., Yao, Y., Liu, Z., Sun, M., Zhang, X., & Lin, L. (2021). UPRec: User-Aware Pre-training for Recommender Systems. Retrieved from arXiv:2102.10989 [cs.IR]

【3】Yuan, F., He, X., Karatzoglou, A., & Zhang, L. (Year). Parameter-Efficient Transfer from Sequential Behaviors for User Modeling and Recommendation. Shenzhen, China: Tencent. Hefei, China: University of Science and Technology of China. London, UK: Google.

【4】U-BERT: Pre-training user representations for improved recommendation, AAAI 2021

【5】UserBERT: Self-supervised User Representation Learning , arxiv 2021

【6】One4all User Representation for Recommender Systems in E-commerce , arxiv 2021

【7】One Person, One Model, One World: Learning Continual User Representation without Forgetting, SIGIR 2021

【8】Scaling Law for Recommendation Models: Towards General-purpose User Representations , AAAI 2023

【9】Self-supervised Learning for Large-scale Item Recommendations , CIKM 2021

【10】IntTower: the Next Generation of Two-Tower Model for Pre-Ranking System, CIKM 2022

【11】Language Models as Recommender Systems: Evaluations and Limitations , NeurIPS 2021 Workshop ICBINB

【12】CTR-BERT: Cost-effective knowledge distillation for billion-parameter teacher models

【13】Recommendation as Language Processing (RLP): A Unified Pretrain, Personalized Prompt & Predict Paradigm (P5) , Recsys 2022

【14】PTab: Using the Pre-trained Language Model for Modeling Tabular Data, arxiv 2022

【15】Is ChatGPT a Good Recommender A Preliminary Study, arxiv 2023

【17】Uncovering ChatGPT’ s Capabilities in Recommender Systems, arxiv 2023

【18】Recommendation as Instruction Following: A Large Language Model Empowered Recommendation Approach, arxiv 2023

【19】Self-supervised Graph Learning for Recommendation , SIGIR 2021

【20】Self-Supervised Hypergraph Convolutional Networks for Session-based Recommendation , AAAI 2021

【21】Where to Go Next for Recommender Systems? ID-vs. Modality-based recommender models revisited, SIGIR 2023

【22】Tenrec: A Large-scale Multipurpose Benchmark Dataset for Recommender Systems, NeurIPS 2022

【23】How Can Recommender Systems Benefit from Large Language Models: A Survey, arxiv 2023

【24】1."Neural Collaborative Filtering" by Xiangnan He, et al. (WWW 2017)

【25】Deep Interest Network for Click-Through Rate Prediction" by Guorui Zhou, et al. (KDD 2018)

【26】Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.